OP864

Prior Art Reference:

Japanese Patent Laid-Open Publication No. Hei 1-258029
Laid-Open Date: October 16, 1989
Title of the Invention: DATA PROCESSING SYSTEM
Filing No. Sho 63-85006
Filing Date: April 8, 1988
Inventor: Shinobu KOIZUMI
c/o Kabushiki Kaisha Hitachi Seisakusho,
System Development and Research Institute
Kawasaki-shi, Kanagawa-ken, Japan
Applicant: Kabushiki Kaisha Hitachi Seisakusho
Chiyoda-ku, Tokyo, Japan

Partinent Description (page 220, upper right column):

[Prior Art]

For linking and editing a plurality of separately prepared programs into one program by linking them one another, there are roughly the following two methods.

A first method is to process a plurality of prepared programs, prior to executing them, by a linking and editing program called a linkage editor. Hereinafter, this first method will be reffered to as a static linkage system. On the other hand, a second method is to link and edit the plurality of programs, during execution thereof, as shown in, for example, Madnick, S.E., Donovan, J.J. "Operating Systems", McGraw-Hill, 1974, pp173-179. Hereinafter, this second method will be referred to as a dynamic linkage system.

⑲日本国特許庁（JP）　⑪特許出願公開

⑫公開特許公報（A）　平1－258029

⑤1Int. Cl.⁴	識別記号	庁内整理番号	⑬公開　平成1年(1989)10月16日
G 06 F 9/06	410	E－7361－5B	

審査請求　未請求　請求項の数 5　（全7頁）

⑤4発明の名称　データ処理装置

⑳特　願　昭63－85006
㉒出　願　昭63(1988)4月8日

⑦2発明者　小泉　忍　神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内
⑦1出願人　株式会社日立製作所　東京都千代田区神田駿河台4丁目6番地
⑦4代理人　弁理士　小川　勝男　外1名

明　細　書

1．発明の名称

データ処理装置

2．特許請求の範囲

1．プログラムとデータが格納される記憶装置と中央処理装置とを備え、複数のプログラムとそれぞれのプログラムが使用するデータ領域とをそれぞれに識別名を付与して上記記憶装置に格納しておき、プログラムの呼び出しとデータ領域へのアクセスに上記記憶装置上のそれぞれの格納番地を用いるデータ処理装置において、記憶装置上のすべてのプログラムとデータ領域についてそれぞれの識別名を示す記号列と記憶装置上の格納番地との対応表を具備し、上記データ処理装置が取り扱うデータとしてプログラムまたはデータ領域の記憶装置上の格納番地とそれらの識別名を示す記号列とを対にしたデータ形式と前記データ形式中の記憶装置上の格納番地を保持するフイールドについてそれぞれが未設定であるという意味の値とを有し、上記データ形式を用いてプログラムを呼び出す命令および上記データ形式を用いてデータ領域にアクセスする命令とを具備し、それら命令の実行時に前記データ形式中の記憶装置上の格納番地を保持すべきフイールドが未設定である場合に前記データ形式中のプログラムまたはデータ領域の識別名を示す記号列を保持しているフイールドの値と前記対応表とを用いてプログラムまたはデータ領域の格納番地を得てプログラムを実行することを特徴とするデータ処理装置．

2．請求範囲第1項記載のデータ処理装置において、プログラムまたはデータ領域の識別名を示す記号列とそれらと記憶装置上の格納番地との対応表により得られたプログラムまたはデータ領域の記憶装置上の格納番地前記データ形式中の記憶装置上の格納番地を保持すべきフイールドに設定することにより、命令の2回目以後の実行に際して前記対応表の検索を省略することを特徴とするデータ処理装置．

3．プログラムとデータを格納する記憶装置とプ

ログラムを実行する中央処理装置とを備え、複数のサブプログラムとそれらが使用するデータ領域とで一つのモジュールを構成し、複数のモジュールを結合することにより全体プログラムを構成するデータ処理装置において、モジュールの結合を機械語1命令で実行することを特徴とするデータ処理装置。

4．請求の範囲第3項記載のデータ処理装置において、あるモジュール中のサブプログラムから別のモジュール中のサブプログラムの呼び出しとそれらのモジュールの結合を同時に行なう機械命令を具備する情報データ処理装置。

5．請求範囲第3項記載のデータ処理装置において、あるモジュール中のサブプログラムから別のモジュールのデータ領域の参照とそれらモジュールの結合を同時に行なう機械命令を具備するデータ処理装置。

3．発明の詳細な説明

〔産業上の利用分野〕

本発明は、データ処理装置のプログラムを複数に分けて作成した際にそれらをプログラムの実行中に結合編集するデータ処理装置に関するものである。

〔従来の技術〕

別々に作成された複数のプログラムを相互に結合しひとつにまとめる結合編集には、大別して次の2つの方法がある。

その第1は、複数の作成済プログラムを、それらの実行に先立ちリンケージエディタと呼ぶ結合編集プログラムにより処理する方法である。以下では、この第1の方法を静的結合方式と呼ぶ。これに対し第2の方法は、例えばMadnick,S.E., Donovan,J.J. “Operating Systams”, McGrav-HiLL, 1974 pp173－179, に示すように、複数のプログラムを、それらの実行中に結合編集する方法である。以下では、この第2の方法を動的結合方式と呼ぶ。

〔発明が解決しようとする課題〕

動的結合方式に比べ静的結合方式には本質的に次の様な欠点がある。

(1) 結合編集を行なつた複数のプログラムのうちのひとつを修正すると全体を再度結合編集しなおさなければならない。

(2) 結合編集されてできたプログラムのひとつのまとまりと、同様に結合編集されてできたプログラムの別のまとまりの中に同一のプログラムが別々に含まれていることがある。

プログラムの大型化・複雑化に伴ない、上記の様な欠点のない動的結合方式が望まれるが、従来の動的結合方式では結合編集されるプログラム中に第4図のフローチャートで示した様な動的結合処理、すなわち、実際に結合先プログラムを呼び出す時点で、プログラム自身が結合編集済かどうかを判定し、（ステップ41）結合編集済ならば記憶領域から呼び先プログラムの入り口番地を取り出し分岐するか、（ステップ42，43）結合編集未完ならば呼び先の検索と分岐を管理プログラム（オペレーティングシステム）に依頼するためのＬＩＮＫマクロを発行する（ステップ44）というコードを組み込んでおく必要があつた。

また、汎用的なある機能を提供するモジュールがありこれを複数の異なるプログラムで利用する場合、上記静的結合方式では、各プログラムの実行前に結合を完了し全体を完成させるため、汎用機能提供モジュールのコピーを作成しそれぞれのプログラムに結合していた。この方法では、プログラムを格納する記憶装置の必要容量が増大するばかりでなく、汎用機能提供モジュールの修正の手間の増大という問題も生じる。一方従来の動的結合方式では、結合の基本単位がサブプログラムであるため、データとそれに対する複数の操作をカプセル化するデータ抽象化の手法に合致しない面があつた。

本発明の第1の目的は、上記の様に動的結合処理を個々のプログラムに組み込むことなく実行中にプログラムの結合編集を行なうことにより、動的結合方式に対して、静的結合方式と同様の記述の簡便さと実行時の高速性を与えるデータ処理装置を提供することにある。

本発明の第2の目的は、モジュールすなわちデ

ータ領域と複数のサブプログラムを単位として動的結合可能とすることにより、実行前に結合する場合に生じるモジュールの複数コピーの問題を回避できるデータ処理装置を提供することにある。

〔課題を解決するための手段〕

上記目的を達成するために本発明の第１の特徴は、

（ａ１）結合編集が未完了である状態を示す値（以下では、この値を未結合値と呼ぶ）かまたは結合先の番地を保持するフィールド（以下では、このフィールドをＡＤＲＦ部と呼ぶ）と、結合先の名前を示す記号列を保持しているフィールド（以下では、このフィールドをＮＡＭＦ部と呼ぶ。）との２つのフィールドを対にしたデータ形式のデータ。

（ｂ２）上記（ａ１）のデータ形式のデータを参照してプログラムの呼び出しを行なう命令（以下では、この命令をＣＡＬＬ命令と呼ぶ）、及び上記（ａ１）のデータ形式のデータを参照してデータ領域へアクセスする命令（以下では、この命令をＤＡＣＳ命令と呼ぶ）。

（ｃ１）記憶装置上のすべてのプログラムやデータ領域の番地を、それらの名前を示す記号列とともに記録してあるテーブル（以下では、このテーブルをＭＡＰテーブルと呼ぶ）。

（ｄ１）上記（ｂ）の命令の実行時に上記（ａ１）のデータと上記（ｃ１）のテーブルを利用して結合編集を行なう機構（以下では、この機構をLINK機構と呼ぶ）。

を備えたことにある。

上記目的を達成するために、本発明の第２の特徴は、モジュールを再入可能にし、かつ、プログラム実行中に結合可能にすることにあり、そのため以下のものを備える。

（ａ２）外部モジュール参照のための命令、すなわち外部モジュール中のサブプログラムの呼び出しと外部モジュールのデータ領域中のデータにアクセスするための命令群。

（ｂ２）起動済モジュール管理テーブル；以下の情報の並びである。

１．起動済モジュールの名称を示す記号列。
２．起動済モジュールのコード領域の先頭番地。
３．起動済モジュールのデータ領域の先頭番地。

なお、本テーブルはタスク毎に作成する。

（ｃ２）上記（ａ２）の命令のオペランドとなるデータ形式：以下の情報フィールドを含む。

１．参照先モジュールの名称を示す記号列。
２．サブプログラム呼び出し命令のオペランドの場合モジュールのコード領域先頭からサブプログラム入口点までのオフセット。データアクセス命令の場合モジュールのデータ領域先頭からそのデータまでのオフセット。
３．結合完了以前には未結合を示す値、結合完了後はそれぞれ目標とするサブプログラム入口点の番地またはデータの番地。

（ｄ２）モジュール管理・結合機構。

〔作用〕

上記（ｂ１）のＣＡＬＬ命令とＤＡＣＳ命令はその実行中に上記（ａ１）のデータ形式のデータを参照する。ここで上記（ａ１）のＡＤＲＦ部の値が未結合値の場合中央処理装置は上記（ｄ１）のLINK機構を起動する。ＬＩＮＫ機構は、上記（ａ１）のＮＡＭＦ部の値である記号列を上記（ｃ１）のMAPテーブル中より検索し対応するプログラムまたはデータ領域の記憶装置上の番地を得て（ａ１）のＡＤＲＦ部に設定する。さらに中央処理装置はＬＩＮＫ機構が得た番地によりＣＡＬＬ命令、ＤＡＣＳ命令の実行を続行する。

また上記において上記（ａ１）のＡＤＲＦ部の値が未結合値でなく正しい番地が設定されていれば、中央処理装置はＬＩＮＫ機構を起動することなくＣＡＬＬ命令ＤＡＣＳ命令を実行する。

以上の構成及び動作によれば、結合編集される個々のプログラムは、上記（ａ１）のデータ形式と上記ｂの命令のみを用いることで、そのプログラム中に直接動的結合処理を組み込むことなくその実行中に結合編集を行なうことができる。

また、上記（ｃ２）の３のフィールドには初期状態において未結合を示す値を設定する。中央処理装置は上記（ａ２）の命令実行時に、その命令のオ

ペランド中の上記（ｃ２）－３のフイールドが未結合を示す値かどうかをチエツクし、未結合の値であれば上記（ｄ２）のモジユール管理・結合機構を起動する。モジユール管理・結合機構は上記(c2)－１のモジユール名称の記号列と上記（ｂ２）の起動済モジユール管理テーブルによりモジユールのコード領域先頭番地またはデータ領域先頭番地を検索し、上記（ｃ２）－２のオフセツト値を加算して目標の番地を決定後上記（ｃ２）－３のフイールドに設定した後、所定の命令動作に復帰する。

以上の動作により、命令の実行時にモジユールの結合が完了する。

〔実施例〕

以下、本発明の一実施例を第１図により説明する。

データ処理装置は、中央処理装置１、記憶装置２とそれらを結合するシステムバス３から構成される。記憶装置２の上には、結合編集されるべき２つのプログラム（ＭＡＩＮプログラム１０とＳＵＢプログラム１８）とＭＡＩＮプログラムが使用するＭＡＩＮデータ領域１４が置かれている。ＭＡＩＮプログラム１０中にはＳＵＢプログラム１８を呼び出すためのＣＡＬＬ命令１１がある。このＣＡＬＬ命令１１は、命令コード１２とオペランド部１３からなるが、命令コード１２は中央処理装置１の内部にある命令実行機構４に対する指令となつており、オペランド部１３は命令実行の際のデータの位置を示すものである。ここではＣＡＬＬ命令１１のオペランド部１２に、MAINデータ領域１４中のリンクデータ１５の番地が設定されている。リンクデータ１５は、呼び先であるＳＵＢプログラム１８の番地を保持する予定のＡＤＲＦ部１６と、ＳＵＢプログラム１８の名前記号列が設定されているＮＡＭＦ部１７から構成されている。ここでＣＡＬＬ命令１１を一度も実行していない状態では、ＡＤＲＦ部１６には番地が未設定であることを示す値＝未結合値が設定されている。

一方、中央処理装置１には命令実行機構４があり、さらにその中にＬＩＮＫ機構５と内部バス６を通じて接続されるＭＡＰテーブル７がある。ＭＡＰテーブル７は、記憶装置２の上にあるすべてのプログラムやデータ領域の名前記号列を保持している名前記号部８と、それぞれのプログラムやデータ領域の記憶装置２上の番地を記録しているロード番地部９から構成されている。

さて、ここでＭＡＩＮプログラム１０中のCALL命令１１をはじめて実行すると、命令実行機構４が命令コード１２を解析し、これがプログラム呼び出し命令であると解釈し第２図のフローチヤートに示す処理を命令実行機構４が行なう。すなわち、この場合、ＣＡＬＬ命令１１のオペランド部１３が示すリンクデータ１５のＡＤＲＦ部１６は初期状態で未結合値であるので命令実行機構１４中のＬＩＮＫ機構５が起動される。（ステツプ２３）ＬＩＮＫ機構５は、ＣＡＬＬ命令１１のオペランド部１３が示すリンクデータ１５とＭＡＰテーブル７を用いて第３図のフローチヤートに示す処理を行なう。すなわち、ＭＡＰテーブル７の名前記号部８の名前記号列の中からリンクデータ１５のＮＡＭＦ部１７の名前記号列と一致するものを検索し（ステツプ３２）、ＭＡＰテーブル７のロード番地部９からその名前記号列に対応するものの記憶装置２上の番地を取り出し（ステツプ３３）、その番地をリンクデータ１５のＡＤＲＦ部に設定すると同時に命令実行機構４に対しその番地を通知してＬＩＮＫ機構５の動作を終了する。これにより命令実行機構４はＣＡＬＬ命令１１のプログラム呼び出し先であるＳＵＢプログラム１８の番地を知り、プログラム呼び出し処理を続行する。

以上の様にして、ＣＡＬＬ命令１１の実行中に呼び先であるＳＵＢプログラム１８との結合編集が行なわれる。

一方、上記の様にリンクデータ１５のＡＤＲＦ部１６にＳＵＢプログラム１８の番地が設定された後、ＣＡＬＬ命令１１を実行すると、第２図のフローチャートが示すようにＬＩＮＫ機構５が起動されることなく呼び出しが行なわれる。

この例ではプログラムの呼び出しについて記述

したが、データ領域へのアクセスの場合もLINK機構の動作については同様であり、単にデータアクセス命令の最終的な動作のみが異なる。

また、この例ではＬＩＮＫ機構５とＭＡＰテーブル７を命令実行機構の中に置いているが、これらを記憶装置２上に置き、リンクデータ１５のＡＤＲＦ部１６が未結合値の場合に命令実行機構４が記憶装置２上のＬＩＮＫ機構５を起動する方法もある。

次に、本発明の他の実施例を第５図により説明する。

本図は、ＭＡＩＮモジュール５１から、オペランドデータ５３とＣＡＬＬ命令５８により、SUBMODモジュール６６の中のＳＵＢサブプログラム６９を呼び出すものである。

ＣＡＬＬ命令５８はＭＡＩＮモジュールコード領域５７にあり、命令の種別を示す命令コード部５９とＭＡＩＮモジュールデータ領域５２中のオペランドデータ５３を示すオペランド部６０からなる。一方オペランドデータ５３は、参照先のモジュール名を示す参照先モジュール名称記号列５６、参照先のSUBMODモジュールコード領域６８の先頭から目標とするＳＵＢサブプログラム６９の入口点までのオフセットｍを保持するオフセット部５５、および初期状態において未結合を示す値を保持し結合後は目標であるＳＵＢサブプログラム６９の入口点番地を保持するポインタ部５４からなる。

ＣＡＬＬ命令５８は中央処理装置により実行されるが、この際オペランドポインタ５８が示すオペランドデータ５３のポインタ部５４が未結合値かどうかが中央処理装置によりチエツクされる。もしここで未結合値でなければポインタ部５４が保持している値をサブプログラム入口点番地を解釈して、ただちに呼び出しが行なわれる。一方ポインタ部が未結合値であれば呼び出し動作を中断し、モジュール管理・結合機構を起動する。

モジュール管理・結合機構は第６図に示す処理を行なう。すなわち、起動済モジュール管理テーブル６１において起動済モジュール名称記号列６５とオペランドデータ５３の参照モジュール名称記号列５６とが一致する起動済モジュールデータ６２を検索する（ステツプ７１）。ここで一致する起動済モジュールデータ６２が存在する場合、元の命令がサブプログラムが呼び出し命令であれば（ステツプ７４）コード領域ポインタ６４の値とオペランドデータ５３のオフセット部５５の値を加算しポインタ部５２に設定、また元の命令がデータアクセス命令であればデータ領域ポインタ６３の値とオフセット部５５の値を加算しポインタ部５４に設定しそれぞれの命令動作を再開する。第５図の場合はサブプログラム呼び出し命令なのでコード領域先頭ポインタ６４の値とオフセット部５５の値の和がポインタ部５４に設定されることになる。また、起動済モジュール管理テーブル６１中に参照モジュール名称記号列６６に一致する記号列をもつ起動済モジュールデータ６２が存在しない場合には、新たなデータ領域とコード領域を記憶域中に確保し、コード領域に対応するモジュールのコードをロードし、データ領域の番地とコード領域の番地とモジュール名称を示す記号列をもつ新たな起動済モジュールデータ６２を起動済モジュール管理テーブル６１に追加登録後、オペランドデータ５３のポインタ部５４の値の設定を行なう。

なお、第５図はサブプログラム呼び出しの場合であるが、データアクセスの場合も第５図のオペランドデータ３と同形式のデータを利用することで実行時にモジュールを結合することができる。

〔発明の効果〕

以上の様に、本発明の第１の特徴であるデータ処理装置におけるプログラムの結合編集に動的結合方式を採用する場合に次の２つの効果がある。

(1) 個々のプログラムについて動的結合処理の組み込みを省略できる。

(2) 結合編集が必要な命令の複数回の実行において、結合編集は最初の命令実行時にのみ行なわれ２回目以後は命令実行が高速となる。

また本発明の第２の特徴によれば、モジュールを実行時に結合するため、実行前に結合する場合

に生じるモジュールの複数のコピーの問題を回避することができる。特に、本発明ではモジュールのデータ領域へのポインタとコード領域へのポインタを別々に起動済モジュール管理テーブルに持つので、モジュールの構成要素のうちプログラムの実行により変化しない部分をコード領域とし変化する部分をデータ領域として別々に管理することができる。このため、一つの中央処理装置が時分割によつて並行的に複数のプログラムを実行するマルチタスク環境下において、各タスク（プログラム）毎に起動済モジュール管理テーブルを作成しデータ領域を個別に確保することで、コード領域を全タスクで共用することが可能であり、プログラムの実行に必要な記憶容量が削減できる効果がある。

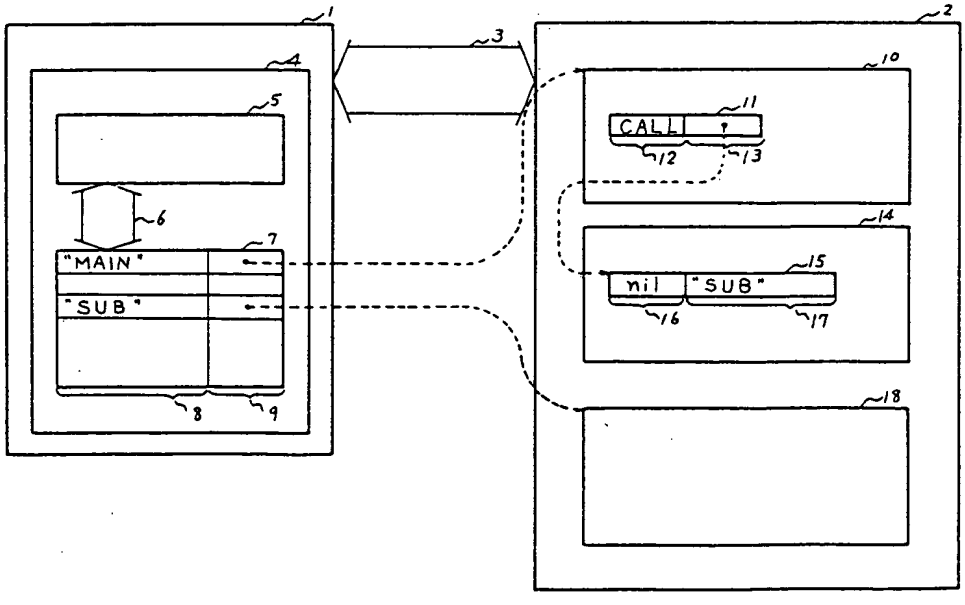
4．図面の簡単な説明

第１図は本発明の一実施例の全体説明図、第２図は第１図中の命令実行機構がＣＡＬＬ命令を実行する時の処理概要を示すフローチャート、第３図は第１図中のＬＩＮＫ機構の動作を示すフローチャート、第４図は従来の動的結合方式においてプログラムに組み込んでいた動的結合処理を示すフローチャート、第５図は本発明の他の実施例の全体説明図、第６図はモジュール管理・結合機構の概略処理の流れ図である。

１…中央処理装置、２…記憶装置、３…システムバス、４…命令実行機構、５…ＬＩＮＫ機構、６…内部バス、７…ＭＡＰテーブル、８…名前記号部、９…ロード番地部、１０…ＭＡＩＮプログラム、１１…ＣＡＬＬ命令、１２…命令コード、１３…オペランド部、１４…ＭＡＺＮデータ領域、１５…リンクデータ、１６…ＡＤＲＦ部、１７…ＮＡＭＦ部、１８…ＳＵＢプログラム。

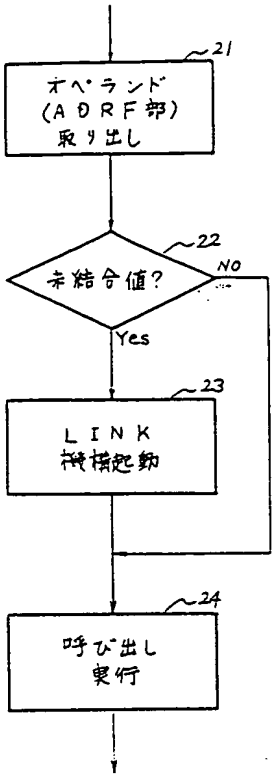
代理人　弁理士　小川勝男

第 1 図

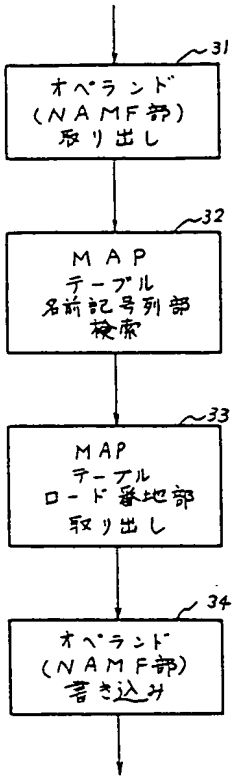


1 中央処理装置　7 MAPテーブル　15 リンクデータ
2 記憶装置　10 MAINプログラム　18 SUBプログラム
5 LINK機構　11 CALL命令

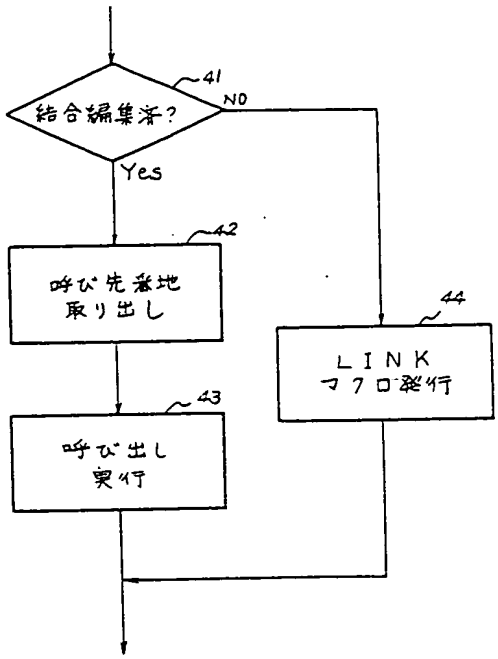
第 2 図



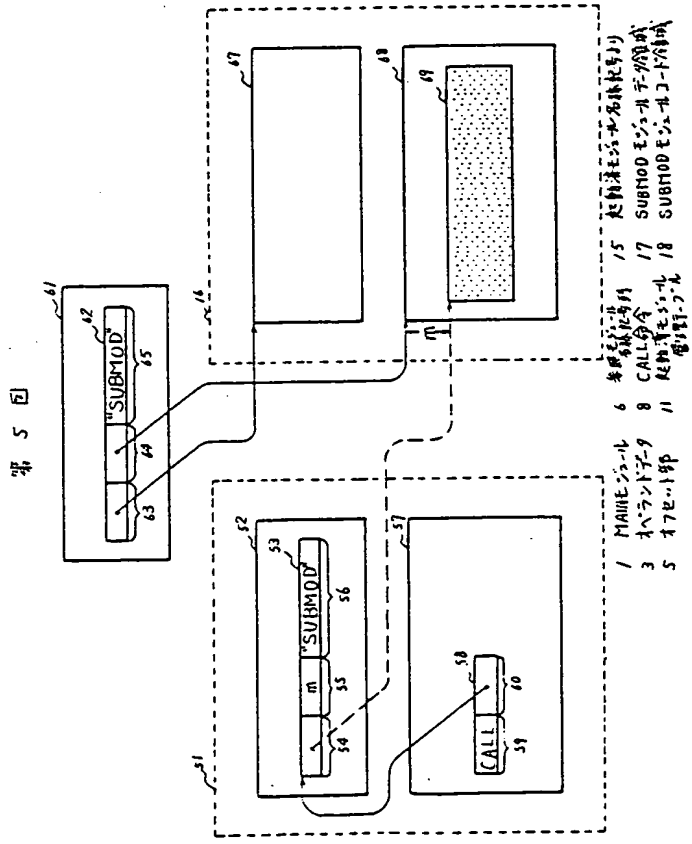
第 3 図



第 4 図



第 5 図



第 6 図